Scalability Analysis and Improvement of Hadoop Virtual Cluster with Cost Consideration

Yanzhang He, Xiaohong Jiang, Zhaohui Wu, Kejiang Ye, Zhongzhong Chen
College of Computer Science, Zhejiang University
Hangzhou, China, 310027
{heyanzhang, jiangxh, wzh, yekejiang, doublezhongchen}@zju.edu.cn

***Abstract*—With the rapid development of big data and cloud computing, big data analytics as a service in the cloud is becoming increasingly popular. More and more individuals and organizations tend to rent virtual cluster to store and analyze data rather than building their own data centers. However, in virtualization environment, whether scaling out using a cluster with more nodes to process big data is better than scaling up by adding more resources to the original virtual machines (VMs) in cluster is not clear. In this paper, we study the scalability performance issues of hadoop virtual cluster with cost consideration. We first present the design and implementation of *VirtualMR* platform which can provide users with scalable hadoop virtual cluster services for the MapReduce based big data analytics. Then we run a series of hadoop benchmarks and real parallel machine learning algorithms to evaluate the scalability performance, including scale-up method and scale-out method. Finally, we integrate our platform with resource monitoring module and propose a system tuner. By analyzing the monitored data, we dynamically adjust the parameters of hadoop framework and virtual machine configuration to improve resource utilization and reduce rent cost. Experimental results show that the scale-up method outperforms the scale-out method for CPU-bound applications, and it is opposite for I/O-bound applications. The results also verify the efficiency of system tuner to increase resource utilization and reduce rent cost.**

Keywords-scalability; MapReduce; cloud computing; big data; rent cost

I. INTRODUCTION

Big data [1] has recently received plentiful attention from academia and industrial community due to the continuous growth of data generated from various fields such as particle physics, biomedical science, earth observation, etc. Regardless of enormous opportunities brought by big data, such as disease prevention, crime combat and real-time roadway traffic query; it also faces challenges in the aspects of data acquisition, storage and analytics [2]. Data-intensive parallel computing is becoming increasingly common, and has been facilitated by frameworks such as Dryad [3] and Google's MapReduce [4] which are paradigms that can scale the big data parallel computing jobs across large-scale cluster of commodity machines. Apache's Hadoop [5] is the open-source implementation of MapReduce which can process hundreds of terabytes of data on at least 10,000 cores.

Cloud is a large shared resource pool of easily usable and accessible storage and computation infrastructures, which can be elastically configured to accommodate various kinds of workloads. These large-scale shared resources are usually exploited by a *pay-as-you-go* model, which means users are only billed for procured storage and computation resources. Virtualization technology (Xen [6], VMware [7], and KVM [8]), as one of the most prominent technologies, provides an abstraction of hardware resources enabling multiple instances of operating system to run simultaneously on a single physical machine (PM). Except for that, VM will be the basic computation unit in the cloud computing environment, and it has the flexibility in dynamic configuration of CPU, memory, disk and network capacity.

When we apply the virtualization technology in big data analytics, there are several challenges to overcome due to the virtualization overheads. For example, to ensure system integrity, the virtualization hypervisor has to trap and process privileged instructions from the guest VMs. This overhead is especially obvious for I/O virtualization and is not favored by data-intensive parallel computing applications where disk access and network communication performance may be critical. Big data analytics which needs to be processed on large-scale distributed platforms in parallel with high efficiency is another big challenge. Despite the above challenges, more and more individuals and organizations will tend to rent virtual clusters in cloud for big data analytics, and virtualization technology is massively used in big data analytics as one of key technologies of cloud computing. There are two reasons: (1) In cloud era, VM will be the basic computation unit. The hadoop virtual cluster can benefit from the advantages: dynamic configuration, server consolidation, ease of management and live migration. (2) Pay-as-you-go model is a utility computing billing method that is adopted in cloud computing. Cloud service user can simply rent a hadoop virtual cluster from Amazon EC2 to run the MapReduce applications without purchasing expensive PMs. In other words, a pay-as-you-go user is just billed for procured storage and computation resources.

How to spend the least money to achieve the required calculation capacity is a main challenge. Whether scaling out using a cluster with more nodes to process big data is better than scaling up by adding more resources to the original virtual machines (VMs) in cluster is not clear. The motivation of our paper is to analyze which scalability method is better of different workloads, and monitor resource utilization for adjusting hadoop or VM configurat-

978-1-4799-5063-8/14 $31.00 © 2014 IEEE
DOI 10.1109/CLOUD.2014.85

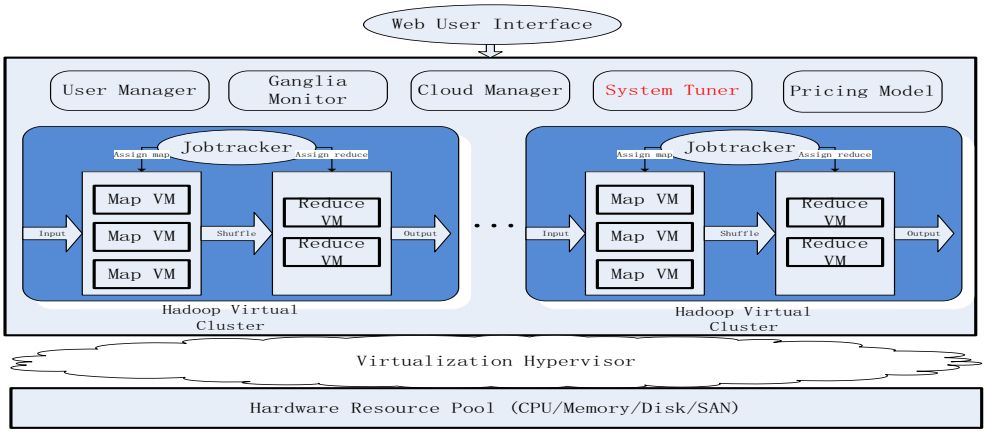

Figure 1. *VirtualMR* Cloud Platform for Big Data Analytics

ion parameters dynamically to reduce rent cost.

In summary, the main contributions of our work are summarized as follows:

- First, we propose and implement a new cloud platform *VirtualMR* which can provide users with scalable hadoop virtual cluster services for MapReduce based big data analytics. User can dynamically adjust the configuration of VMs include CPU, memory and disk capacity or the total VM amount within the cluster through a browser.
- Then we perform a series of experiments to study the scalability performance of hadoop virtual cluster with different kinds of parallel computing applications. Experimental results show that the scale-up method outperforms the scale-out method for CPU-bound applications, and it is opposite for I/O-bound applications.
- Finally, we integrate our platform with resource monitoring module and design a system configuration parameter tuner. By analyzing the monitored data, we adjust the hadoop framework parameters and VM configuration to improve resource utilization and reduce rent cost.

The rest of this paper is organized as follows. In section II, we design and implement the cloud platform *VirtualMR*. In section III, we study the scalability performance of hadoop virtual cluster with different big data applications. In section IV, by monitoring the cluster resource utilization, we propose a system configuration parameter tuner to improve performance and resource utilization and reduce rent cost. In section V, we present the related work. Finally we give our conclusion and future work in section VI.

II. VIRTUALMR PLATFORM AND KEY TECHNOLOGIES

In this section, we first describe the design and implementation of *VirtualMR* platform, and then introduce some key technologies of this platform.

A. Architecture of VirtulMR Platform

Figure 1 illustrates the architecture of *VirtualMR* platform for big data analytics. It consists of four tiers from bottom to top: Hardware resource pool, Virtualization hypervisor, Parallel computing engine and Web user interface. The parallel computing engine tier is the core component in our paper, and it includes six modules: Hadoop virtual cluster, User manager, Ganglia [1] monitor, Cloud manager, System tuner and Pricing model. All the six modules cooperate with each other to provide a scalable hadoop virtual cluster for big data analytics.

Users apply for virtual clusters through browser, and then the web user interface invokes the cloud manager to create VMs upon the hardware resource pool. After the VMs become running, users can automatically deploy (*auto-deploy*) hadoop framework in their own virtual cluster and submit data-intensive parallel computing applications. The newly designed and developed auto-deploy algorithm integrated in cloud manager is responsible for setting the initial configuration files and start the hadoop daemons. The configuration files include: *masters*, *slaves*, *core-site.xml*, *hdfs-site.xml* and *mapred-site.xml*, etc. Using the auto-deploy, cluster nodes can be easily added or removed as needed. For example, to increase the storage capacity and computation capability available to a cluster, we can add new nodes. Conversely, sometimes we may wish to shrink a cluster, we can remove nodes. The current version of hadoop we use is 1.0.4 in VM image.

After data-intensive parallel computing applications running in the cluster, both the Master node and Slave nodes are monitored by Ganglia monitoring software. Ganglia is a scalable distributed monitoring system for high performance computing systems such as clusters and grids based on a hierarchical design. It is responsible for monitoring the resource utilization of all nodes in virtual cluster, including CPU, memory, disk, and network. By analyzing the monitored data, we can find the performance bottlenecks

[1] http://ganglia.sourceforge.net/

and adjust the hadoop framework parameters and VM configuration by the system tuner which is responsible for altering configuration parameters.

B. Key Technologies

1) Hadoop: The hadoop [5] project includes two main modules: (1) Hadoop Distributed File System (HDFS) which provides high-throughput access to application data, and (2) MapReduce which is a YARN-based paradigm for parallel processing of large-scale data sets. The architecture of hadoop is based on Master-Slave mode which may result in single node of failure.

The HDFS module contains three kinds of node: NameNode, DataNode and Client. The NameNode holds the namespace of distributed file system, which is a hierarchy of files and directories. Both file and directory are represented by *inode* which records the attributes like permission, modification, access time and disk space quota, etc. The DataNode is responsible for the storage of file content itself. When the file size exceeds the appointed block size, it will be split into large blocks (typically 64 megabytes) which are independently replicated on multiple DataNodes with three replicas.

A MapReduce job is a unit of task that the user wants to be performed: it consists of the input data, *mapper* function, *reducer* function and job configuration information. There are one *jobtracker*, a number of *tasktrackers*. After the users upload input data to the HDFS and submit some parallel computing jobs, the jobtracker coordinates all the jobs running on the system by scheduling tasks to run on different tasktrackers. Tasktrackers run tasks and send heartbeats with progress report to the jobtracker, which keeps a record of the overall progress of each job. The underlying runtime system automatically parallelizes the computation across large-scale clusters of machines, handles machine failures, and schedules inter-machine communication to make efficient use of the disk and network.

Xen and OpenNebula: There are two basic kinds of virtualization technology: full virtualization and paravirtualization. Full virtualization is a complete simulation of the underlying hardware, while paravirtualization provides a software interface to VMs and the interface is similar but not identical to that of the underlying hardware.

Xen [6] is an x86 virtualization hypervisor which allows multiple commodity operating systems to share conventional hardware in a safe and resource managed fashion, and without sacrificing either performance or functionality. It is achieved by providing an idealized VM abstraction to which operating systems such as Linux, BSD and Windows XP. We currently use Xen 4.1 version in our platform.

With IaaS cloud service growing popularity, a lot of tools and technologies are emerging that can transform an organization's existing infrastructures into a private or hybrid cloud. OpenNebula [9] is an open source cloud manager that deploys virtualized services on both a local pool of resources and cross-domain clouds. It is responsible for providing a uniform and homogeneous view of virtualized resources, managing a VM's full life cycle, supporting configurable resource allocation policies to meet the organization's specific goals and adapting to an organization's flexible resource needs. We currently use OpenNebula 3.8.1 version as the cloud manager in our platform.

2) Scale-up and Scale-out Method: There are two methods to add more resources for a particular application: scale-up and scale-out. Scale-up method means adding more resources to original nodes in a system, typically involving the addition of CPU and memory to a single PM or VM node. Scale-out method means adding more nodes to a system, such as adding a new VM to a data-intensive parallel computing cluster. Hundreds of small commodity machines may be put together in a cluster to obtain aggregate computing capacity that often exceeds the traditional computers with a single processor.

However, there are tradeoffs between the above two methods. Large numbers of nodes means increased management complexity, as well as a more complex programming model and high network communication overheads between nodes; meanwhile, some applications do not lead themselves to a distributed computing model. It's meaningful to find out which is more efficient to fullfil big data analytics on a scale-up virtual cluster (with fewer powerful nodes) or on a scale-out virtual cluster (with more less powerful nodes), especially with the rent cost consideration.

III. Scalability Performance Analysis of Hadoop Virtual Cluster

In this section, we study the scalability performance of hadoop virtual cluster with different big data analytics applications. We mainly compare the performance of the scale-up method and scale-out method. After analyzing the results, cloud users can choose effective scale method to add more resources to his cluster for specific application.

A. Experimental Configuration

1) Hadoop Virtual Cluster Configuration

All the experiments are performed on Dell PowerEdge R720 servers, with 2 Quad-core 64-bit Xeon processors E5-2620 at 2.00GHz and 64GB DRAM. We use Ubuntu 12.04 in Domain 0, and Xen 4.1 as the virtualization hypervisor. Each virtual machine is installed with Ubuntu12.04 as the guest OS with the configuration of 1~4 vCPU and 3.75~15GB vMemory. According to the Amazon EC2 [2] pricing model in Table I, we set the same price of VM with them, but with small time granularity. In order to ensure the data precision, each of the showed experimental results were gained via running benchmarks three times with the same configuration and use the average value.

[2] http://aws.amazon.com/cn/ec2/

TABLE I. AMAZON EC2 PRICING MODEL

VM Type	vCPU	vMemory	Price
m1.xsmall	1	1GB	0.030$/h
m1.small	1	1.7GB	0.060$/h
m1.medium	1	3.75GB	0.120$/h
m1.large	2	7.5GB	0.240$/h
m1.xlarge	4	15GB	0.480$/h

2) Hadoop benchmark

In our experiments, we choose *-put/-get* FsShell commands to test the performance of HDFS and four typical hadoop benchmarks which include the real parallel machine learning algorithms to test the performance of MapReduce. Table II shows the details of benchmarks.

TABLE II. THE DESCRIPTION OF BENCHMARKS

Name	Category	Description
Put/Get	HDFS	Write/read local file to/from the hadoop distributed file system
WordCount	MapReduce	Reads text files and counts how often each word occur
RegexMatch	MapReduce	Used in pattern matching with strings, or string matching
TeraSort	MapReduce	Sorts amount of data as fast as possible, and validates the accuracy of the result
Mahout	MapReduce	Machine learning library, including clustering, classification and recommendation

The *WordCount* benchmark reads text files and counts how often words occur. Each mapper takes a line as input and partitions it into words. It then emits a key/value pair of the word and 1. Each reducer sums the counts for each word and outputs a single key/value with the word and sum.

In *RegexMatch* benchmark, we use *-grep* command which is used for searching plain-text data sets for lines matching a regular expression.

The *TeraSort* benchmark is to sort amount of data as fast as possible. A full TeraSort benchmark consists of the following three steps: (1) Generating the input data via *TeraGen*. (2) Running the actual TeraSort on the input data. (3) Validating the sorted output data via *TeraValidate*.

The *Mahout*[3] library is an open-source machine learning library based on hadoop. Currently mahout supports mainly three kind algorithms: clustering, classification and recommendation. In this paper, we choose some clustering algorithms, including *k-Means clustering*, *Fuzzy k-Means clustering* and *Canopy clustering*, to test the performance of our *VirtualMR* platform.

[3] http://mahout.apache.org/

The k-Means clustering offered by mahout library is an iterative algorithm implemented as a series of MapReduce rounds. It aims to partition n objects into k clusters in which each object belongs to the cluster with the nearest distance, serving as a prototype of the cluster. The input to the algorithm is a set of objects represented as d-dimensional vectors, and an initial set of cluster centers. The Fuzzy k-Means clustering is an improvement of k-Means, while k-Means creates hard clusters where a object belong to only one cluster, Fuzzy k-Means is a more statically formalized method and creates soft clusters where a particular object can belong to more than one cluster with certain probability. The Canopy Clustering is a very simple, fast and accurate method for grouping objects into clusters. Canopy Clustering is often used as an initial step in more rigorous clustering techniques, such as k-Means Clustering.

B. Performance Comparison of Scale-up and Scale-out Method

In the experiments, we create two clusters to compare and analyze the scalability performance of hadoop virtual cluster with cost consideration. In cluster A (scale-up), we create 4 VM nodes (1 Master and 3 Slaves), each node with 4 vCPU and 15GB vMemory, and set map task slot number=8, reduce task slot number=4. In cluster B (scale-out), we create 16 VM nodes (1 Master and 15 Slaves), each node with 1 vCPU and 3.75GB vMemory, and set map task slot number=2, reduce task slot number=1. The total number of vCPU and vMemory is equal in two clusters. According to Table I, we realize the rent cost per hour (or minute) of cluster A and cluster B is equal. When the parallel computing execution time is short, the rent cost will be low.

Figure 2 shows the write/read performance as the file size ranges from 400MB to 2400MB of different clusters. It is obvious that the write/read time is short when the file size is relatively small, and increases gradually as the file size scales. From the Figure 2(a), we can see that the scale-up method has lower performance than scale-out. The reason is that there are more VMs in cluster B which can provide higher cumulative disk and network I/O bandwidth. The I/O contention and interference between task slots in cluster A is more serious. From the Figure 2(b), we can observe the read time increases with the nodes number scales. In read process, the client chooses the nearest DataNodes to read data. The data volume of read process is one-third of write process, so the I/O will not be the bottleneck in read process. The main time is cost by the NameNode to choose which DataNodes to read the block replicas, and more VMs cost more time.

Figure 3(a) represents the WordCount performance when running on different clusters. The input data of WordCount is chosen from the TOEFL reading materials with the file size ranging from 400MB to 2400MB. From the figure, it is obvious that the running time increases as the size of input data scales. Further, the MapReduce performance of scale-out method outperforms scale-up method. It is because the WordCount belongs to I/O-bound application and more

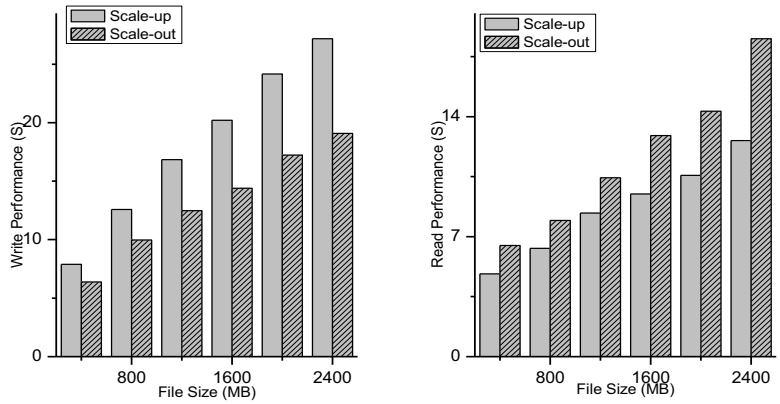

Figure 2. (a) Write Performance of Different Cluster (b) Read Performance of Different Cluster

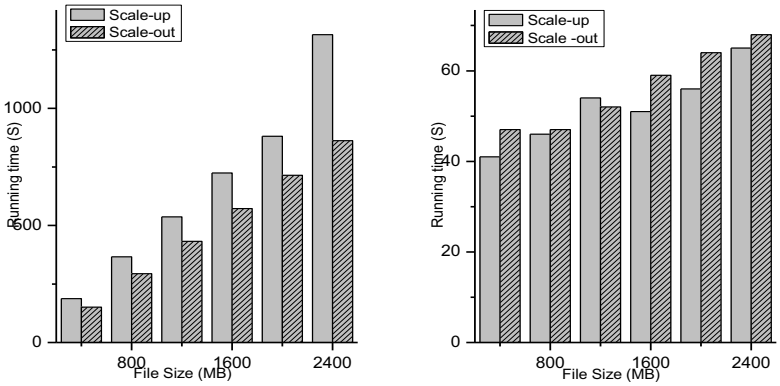

Figure 3. (a) WordCount Performance of Different Cluster (b) Grep Performance of Different Cluster

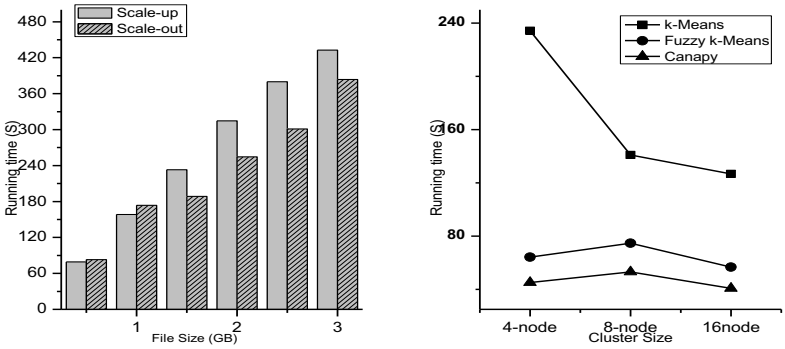

Figure 4. (a) TeraSort Performance of Different Cluster (b) Parallel Clustering on USCensus1990 Data Set of Different Cluster

concurrent nodes provide more total disk and network I/O bandwidth. Generally, I/O-bound refers to a condition in which the time it takes to complete a computation is determined principally by the period spent waiting for input/output operations to be completed.

Figure 3(b) shows the performance of Grep which is CPU-bound application when running on different clusters. An application is called CPU-bound when task execution time is determined principally by the speed of the central processor: processor utilization is high, perhaps at 100% usage for many seconds or minutes. From the figure, we find the performance of scale-up method is better than scale-out method. The reason is that the CPU capability is equal for cluster of A and B, but cluster B needs to transfer more intermediate data in the shuffle phase.

Figure 4(a) illustrates the sort time of TeraSort benchmark which belongs to both I/O-bound and CPU-bound application. It can be used to sort different sizes of input data. In our experiments, we consider the data size of 0.5GB, 1GB, 1.5GB, 2.0GB, 2.5GB and 3GB. From the figure, we can see that the data sort time is nearly proportional with the data size, and the performance of scale-out method outperforms scale-up method which is similar to the phenomenon of Figure 3(a).

Figure 4(b) illustrates the performance of real parallel clustering algorithms, including k-Means, Fuzzy k-Means and Canopy. The data set is the *USCensus1990*[4] which was derived from the USCensus1990raw data set. It includes 68 categorical attributes, and many of the less useful attributes in the original data set have been dropped. Similar to WordCount and TeraSort benchmark, the scale-up method is worse than scale-out method. The reason is that the clustering is an iterative algorithm implemented as a series of MapReduce rounds which causes lots of I/O operations.

Discussion: Since the total disk and network I/O bandwidth of cluster A is lower than cluster B, the file write performance and I/O-bound parallel computing applications has serious I/O contention and interference between each task slot and results in worse performance. For CPU-bound parallel computing applications, we find the performance of scale-up method is better than scale-out method which means the MapReduce performance can be obviously affected by the shuffle process of the intermediate result. The above experiment results merely based on two clusters with fixed configuration, and we will analyze the performance with dynamic adjustment of the hadoop framework parameters and VM configuration in section IV.

IV. CASE STUDY OF SYSTEM TUNER

In this section, we integrate our platform with resource monitoring module and design a system configuration parameter tuner. The monitored data are transferred to the *Utilization Analyzer*, and the analyzer will give some configuration adjustment strategies for us. We choose one strategy and adjust the hadoop framework parameters or VM configuration to reduce job execution time and improve resource utilization. At the same time, we can reduce the rent cost of our clusters. We choose TeraSort benchmark running on cluster B as an example to verify the efficiency of system tuner.

A. Tuning the Hadoop Framwork Parameters

According to the monitored data when TeraSort is running on the hadoop virtual cluster, we observe that the CPU utilization is relatively high and memory utilization is extremely low. Figure 5 shows the performance of tuning map task slot capacity per node and JVM heap size of each map task. We use the system tuner to reset the reduce task slot number=1 and scale the number of map task slot from 3 to 5, and we find that as the number of map task slot scales, the running time changes slightly from Figure 5(a). It is because that the CPU utilization is already high enough and more concurrent task slots will not decrease the running time. The next optimization adjusts the heap size of JVM to be optimal. By default each hadoop map and reduce task run in a JVM with 200MB heap size within which they allocate buffers for in-memory data. When the buffers are full, data is spilled to storage, adding overheads. According to the Ganglia monitor, we note that 128MB per task leaves substantial amounts of memory unused on VMs. In the experiment, we use the system tuner to increase the JVM heap size from 128MB to 512MB, and observe that the performance with 256MB heap size is the best from the Figure 5(b). However, when the heap size reaches 512MB, the performance descends instead due to the garbage collection overheads.

B. Tuning the VM Configuration

In addition to adjusting the hadoop framework parameters, we also can dynamically tune the VM configuration to improve resource utilization and reduce the cluster rent cost with the web user interface. We use the Dynamic Memory Control (DMC) technology to change the amount of host physical memory assigned to any running VM without rebooting it, and use the *xm vcpu-set* command to change the vCPU amount of VM.

Figure 6 shows the TeraSort performance and cluster rent cost with different VM memory. Due to the low utilization of VM memory, we shrink the memory size of each VM from 4GB to 1GB for comparing the performance. From the Figure 6(a), we find the performance variation between different scenarios is relatively small. However, according to Table I, we can calculate the rent cost of clusters with different memory size of VM, and see the reduction is apparent by 68.8% average between 4GB and 1GB from the figure 6(b).

C. Discussion of System Tuner

By running multiple data-intensive parallel computing applications and observing the results of performance change of system tuner, we get some tips for improving the performance of data processing. (1) For CPU-bound applications, if the CPU utilization of the virtual machine is not full, you are suggested to increase the map and reduce task slot capacity per node and decrease the number of nodes in you virtual cluster. (2) For I/O-bound applications, if the I/O utilization is low, you can maximize the disk and network throughput by allocating enough map tasks in each node to access as many files in parallel. (3) For Memory-bound applications, if the memory is not being fully utilized, you may benefit from increasing the JVM heap size or decreasing the total memory of each VM.

V. RELATED WORK

MapReduce and virtualization technology has been widely studied respectively in the recent years. Generally, there are two methods to improve the performance of MapReduce: (1) Adjusting the hadoop framework and job configuration parameters to increase the resource utilization and improve the single job execution performance, and (2) modifying the open source code to avoid some bottlenecks or adapt to some specific applications. *Chen et al.* [10]

[4] http://archive.ics.uci.edu/ml/datasets.html

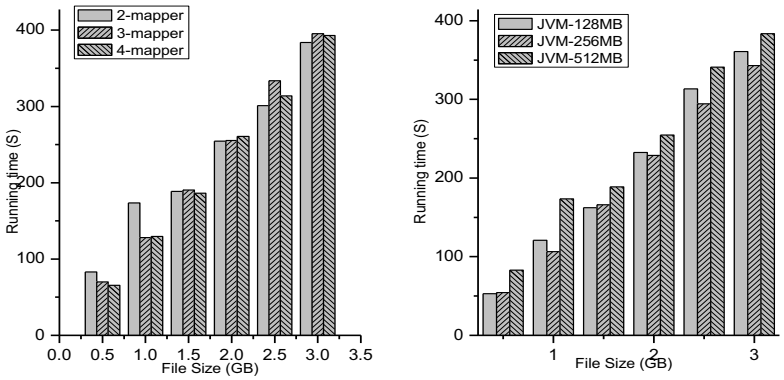

Figure 5. (a) TeraSort Performance with Different Map Task Slot Numder and (b) JVM Heap Size

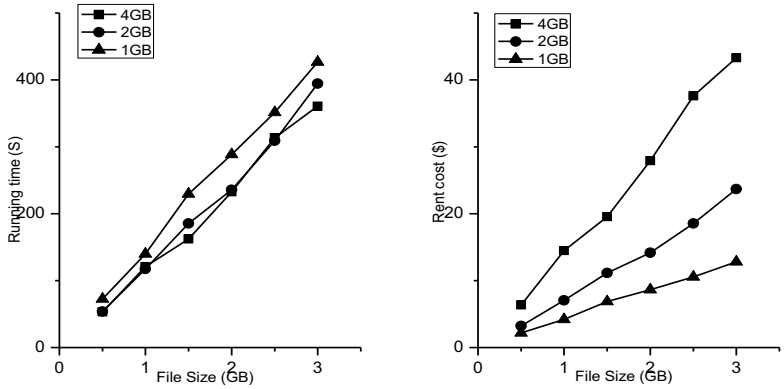

Figure 6. (a) TeraSort Performance and (b) Cluster Rent Cost with Different VM memory

performed multiple experiments on different benchmarks for MapReduce performance evaluations. *Kavulya et al.* [11] analyzed 10-months MapReduce job execution logs from the M45 supercomputing cluster. *Rizvandi et al.* [12] studied CPU utilization time patterns of several MapReduce applications, and used the Dynamic Time Warping (DTW) to predict the probable execution patterns of unknown applications. *Rizvandi et al.* [13] also proposed an analytical method to model the dependency between configuration parameters and total execution time of MapReduce applications. *Lin et al.* [14] constructed models to describe the effects of task granularity, slot count and workload type on the performance from both the single job and system perspectives. The papers [15-16] introduced some self-tuning systems for big data analytics, and our work is a little similar to them, but they have not taken the VM configuration adjustment into consideration.

All the above works focus on PM cluster, and there are also some studies of MapReduce on virtual cluster. *Ibrahim et al.* [17] conducted a series of experiments to measure and compare the performance of hadoop on VMs and PMs. *Ye et al.* [18] proposed the vHadoop which is a scalable hadoop virtual cluster platform for MapReduce based parallel machine learning. *Zhang et al.* [19] used the hadoop framework to develop a user application to process scientific data on clouds. Amazon provides a cloud service of EMR[5] to individuals and organizations to process big data. To reduce the contention and interference of disk access and network communication among VMs, *Fang et al.* [20] explored the relationship between I/O scheduling in a virtualization hypervisor and performance of MapReduce applications running on the VMs. *Geng et al.* [21] defined metrics to analyze the data allocation problem in virtual environment theoretically and designed a location-aware file block allocation strategy which retains compatibility with the native hadoop to reduce data transfer between nodes.

However, most of the previous work has not taken the scalability method and the incurred cost into consideration when running parallel computing applications in cloud. In cloud computing era, pay-as-you-go model is popular billing method. Due to the current market trading mechanism is inflexible, and the price is not reasonable enough in some situation, *Shang et al.* [22] defined a double auction Bayesian Game-based pricing model for the suggested cloud market and discussed how to develop an optimal pricing strategy for this model. *Kambatla et al.* [23] studied the ability to provide dynamical resource provision of MapReduce based applications to minimize the rent cost, while obtaining the best performance.

[5] http://aws.amazon.com/cn/elasticmapreduce/

VI. CONCLUSION AND FUTURE WORK

With the rapid development of big data and cloud computing, more and more individuals and organizations tend to rent virtual cluster in clouds for big data analytics. This paper focuses on the scalability performance analysis with cost consideration, and constructs a cloud platform *VirtualMR* which can provide users with scalable virtual cluster for the MapReduce based big data analytics. We first present the design and implementation of *VirtualMR* platform. Then we perform a series of hadoop benchmarks and real parallel machine learning algorithms to analyze the scalability performance, including scale-up method and scale-out method. Finally, we integrate our platform with resource monitoring and propose a system configuration parameter tuner. By analyzing the monitored data, we can dynamically adjust the hadoop framework parameters and VM configuration to improve resource utilization and reduce cluster rent cost. Experimental results show that: (1) the disk and network I/O are main bottlenecks of *VirtualMR* platform due to the shared resource contention and interference; (2) the scale-up method outperforms the scale-out method for CPU-bound applications, and it is opposite for IO-bound applications; (3) by tuning the system parameters, we can increase resource utilization and reduce rent cost to a large extent. For example, we can reduce rent cost by 68.8% through shrinking the memory size from 4GB to 1GB for TeraSort benchmark.

Future work will include analyzing more characteristics of different kinds of MapReduce job, and creating an machine learning based automatic system tuner which can dynamically change the hadoop framework parameters and VM configuration of data-intensive parallel computing applications (I/O-bound, CPU-bound and Memory-bound).

ACKNOWLEDGMENT

This work is supported by National High Technology Research 863 Major Program of China (NO.2011AA01 A207) and National Natural Science Foundation of China (NO.61272128)

REFERENCES

[1] C. Lynch, “Big data: How do your data grow?,” *Nature*, vlo. 455, no. 7209, 2008, pp.28-29.

[2] A. Labrinidis and H. V. Jagadish, “Challenges and opportunities with big data,” in *Proceedings of the VLDB Endowment* , 2012, pp.2032-2033.

[3] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, “Dryad: distributed data-parallel programs from sequential building blocks,” *ACM SIGOPS Operating Systems Review*, vol. 41, no. 3, 2007, pp.59-72.

[4] J. Dean, and S. Ghemawat, “MapReduce: simplified data processing on large clusters,” *Communications of the ACM*, vol. 51, no. 1, 2008, pp.107-113.

[5] T. White, *Hadoop: the definitive guide*. O'Reilly, 2012.

[6] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, ... and A. Warfield, “Xen and the art of virtualization,” *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, 2003, pp.164-177.

[7] C. A. Waldspurger, “Memory resource management in VMware ESX server,” *ACM SIGOPS Operating Systems Review*, vol. *36*, 2002, pp.181-194.

[8] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, “KVM: the Linux virtual machine monitor,” in *Proceedings of the Linux Symposium* , Vol. 1, 2007, pp.225-230.

[9] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster, “Virtual infrastructure management in private and hybrid clouds,” *Internet Computing*, 13(5), 2009, pp.14-22.

[10] Y. Chen, A. Ganapathi, R. Griffith, and R. Katz, “The case for evaluating MapReduce performance using workload suites,” in *Proceedings of IEEE 19th International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2011, pp.390-399.

[11] S. Kavulya, J. Tan, R. Gandhi, and P. Narasimhan, “An analysis of traces from a production mapreduce cluster,” in *Proceedings of 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid)*, 2010, pp.94-103.

[12] N. Babaii Rizvandi, J. Taheri, and A. Y. Zomaya, “On using pattern matching algorithms in mapreduce applications,” in *Proceedings of IEEE 9th International Symposium on Parallel and Distributed Processing with Applications (ISPA)*, 2011, pp.75-80.

[13] N. Babaii Rizvandi, A. Y. Zomaya, A. Javadzadeh Boloori, and J. Taheri, “On Modeling Dependency between MapReduce Configuration Parameters and Total Execution Time,” 2012.

[14] L. Yin, C. Lin, and F. Y. Ren, "Analysis and Improvement of Makespan and Utilization for MapReduce."

[15] H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. B. Cetin, and S. Babu, “Starfish: A Self-tuning System for Big Data Analytics,” InCIDR, Vol. 11, 2011, pp.261-272.

[16] S. Babu, “Towards automatic optimization of MapReduce programs,” in *Proceedings of the 1st ACM symposium on Cloud computing*, 2010, pp.137-142.

[17] S. Ibrahim, H. Jin, L. Lu, L. Qi, S. Wu, and X. Shi, “Evaluating mapreduce on virtual machines: The hadoop case,” in *Cloud Computing*, 2009, pp.519-528.

[18] K. Ye, X. Jiang,Y. He, X. Li, H. Yan, and P. Huang, “vHadoop: A Scalable Hadoop Virtual Cluster Platform for MapReduce-Based Parallel Machine Learning with Performance Consideration,” in *Proceedings of IEEE International Cenference on Cluster Computing Workshops (CLUSTER WORKSHOPS)*, 2012, pp.152-160.

[19] C. Zhang, H. De Sterck, A. Aboulnaga, H. Djambazian, and R. Sladek, “Case study of scientific data processing on a cloud using hadoop,” in *High performance computing systems and applications*, 2010, pp.400-415.

[20] J. Fang, S. Yang, W. Zhou, and H. Song, “Evaluating I/O scheduler in virtual machines for mapreduce application,” in *Proceedings of 9th International Conference on Grid and Cooperative Computing (GCC)*, 2010, pp.64-69.

[21] Y. Geng, S. Chen, Y. Wu, R. Wu, G. Yang, and W. Zheng, W. , “Location-aware mapreduce in virtual cloud,” in *Proceedings of International Conference on Parallel Processing (ICPP)*, 2011, pp. 275-284.

[22] Q. Zhu, and G. Agrawal, “Resource provisioning with budget constraints for adaptive applications in cloud environments,” in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, 2010, pp.304-307.

[23] K. Kambatla, A. Pathak, and H. Pucha, Towards optimizing hadoop provisioning in the cloud,” in *Proceedings of the First Workshop on Hot Topics in Cloud Computing*, 2009, pp. 118.